1. Consider a 2-dimensinal array as follows: [20%]
   char A[128][128];
   where A[0][0] is stored at address 128, in a paged memory system with pages of size 128 bytes. A small process resides in page 0 (addresses 0 to 127) for manipulating the A matrix; thus every instruction will be fetched from page 0.
   I. For a 4-frame system, how many page faults are generated by the following two array-initialization loop codes, using LRU replacement policy, and assuming frame 0 has the process in it, the other two three are initially empty.
      a.
```
for (j=0; j<128; j++)
    for (i=0; i<128; i++)
        A[i][j] = 0;
```
      b.
```
for (i=0; i<128; i++)
    for (j=0; j<128; j++)
        A[i][j] = 0;
```
   II. Assume that CPU has a 128-block cache, each of which is 32 bytes. The cache is set associative where each set contains 2 cache blocks. How many cache hits and misses are generated for the above two initialization codes? (also assume LRU, cache is initially empty, and physical and logical memory addresses are the same)
   III. What are cold miss and capacity miss?

2. How to create a process that runs a separate program in your program on Unix? Please write a sample program and explain the details of the process creation procedure. How does the copy-on-write virtual memory technique enhance performance of creating and running processes? [15%]

3. What are the necessary conditions for deadlocks? What is the logic that can be used these conditions to develop deadlock free algorithms? What are deadlock prevention and deadlock avoidance algorithms? [15%]

4. Since assembly language is the interface to higher-level software, the assembler can also treat common variations of machine language instructions as if they were instructions in their own right. However, these instructions need not be implemented in hardware. Such instructions are called pseudoinstructions. And many such instruction sets appear in MIPS programs.
   For each pseudoinstruction in the following table, produce a minimal sequence of actual MIPS instructions to accomplish the same thing. You may need to use **$at** for some of the sequences. In the following table, **big** refers to a specific number that requires 32 bits to represent and **small** to a number that can be expressed using 16 bits. [15%]

Pseudoinstruction	What it accomplishes	Solution
move $t5, $t3	$t5 = $t3	(ex.) add $t5, $t3, $zero
clear $t5	$t5 = 0	
li $t5, small	$t5 = small	
li $t5, big	$t5 = big	
lw $t5, big($t3)	$t5 = Memory[$t3 + big]	
addi $t5, $t3, big	$t5 = $t3 + big	
beq $t5, small, L	if ($t5 = small) go to L	
beq $t5, big, L	if ($t5 = big) go to L	
ble $t5, $t3, L	if ($t5 <= $t3) go to L	
bgt $t5, $t3, L	if ($t5 > $t3) go to L	
bge $t5, $t3, L	if ($t5 >= $t3) go to L	

5. To add an addressing mode to MIPS, that allows arithmetic instructions to directly access memory. If we add an instruction, **addm**, as is found in the 80x86, as following brief description:

   addm $t2, 100($t3) # $t2 = $t2 + Memory[$t3+100]

   then please describe the steps of this instruction '**addm**' might take. [10%]
   Then write a paragraph or two explaining why it would be hard to add this instruction to the MIPS pipeline. (Hint: You may have to add one or more additional stages to the pipeline.) [10%]

6. Here are two different I/O systems intended for use in transaction processing:
   - System A can support 1000 I/O operations per second.
   - System B can support 750 I/O operations per second.

   The latency of an I/O operation for these two systems differs. The latency for an I/O on system A is equal to 20 ms, while for system B the latency is 18 ms for the first 500 I/Os per second and 25 ms per I/O for each I/O between 500 and 750 I/Os per second. In the workload, every 10^{th} transaction depends on the immediately preceding transaction and must wait for its completion. What is the maximum transaction rate that still allows every transaction to complete in 1 second and does not exceed the I/O bandwidth of the machine ? (For simplicity, assume that all transaction requests arrive at the beginning of a 1-second interval.) [15%]